What does it mean to build software?

- How many of you know how to **build** software?
  - Really.
What does it mean to build software?

- How many of you know how to build software?
  - Really.

- What does the process include?
What does it mean to build software?

- How many of you know how to build software?
  - Really.

- What does the process include?

- How many of you have heard terms like
  - Build Engineering?
  - Build Configuration?
  - Build Automation?
  - Dependency Management?
  - Continuous Integration?
What does it mean to build software?

- How many of you know how to *build* software?
  - Really.

- What does the process include?

- How many of you have heard terms like
  - Build Engineering?
  - Build Configuration?
  - Build Automation?
  - Dependency Management?
  - Continuous Integration?

*Just getting something to compile reproducibly can be nontrivial*
What does it mean to build software?

- Building software includes (at least):
  - version control integration
What does it mean to build software?

- Building software includes (at least):
  - version control integration
  - identifying dependencies & their versions
What does it mean to build software?

- Building software includes (at least):
  - version control integration
  - identifying dependencies & their versions
  - configuring build commands for different build modes & environments
What does it mean to build software?

- **Building software includes (at least):**
  - version control integration
  - identifying dependencies & their versions
  - configuring build commands for different build modes & environments
  - writing instructions for how to configure & build
What does it mean to build software?

- Building software includes (at least):
  - version control integration
  - identifying dependencies & their versions
  - configuring build commands for different build modes & environments
  - writing instructions for how to configure & build
  - test configuration & execution (performance & correctness)
What does it mean to build software?

- Building software includes (at least):
  - version control integration
  - identifying dependencies & their versions
  - configuring build commands for different build modes & environments
  - writing instructions for how to configure & build
  - test configuration & execution (performance & correctness)
  - automated code quality checking
What does it mean to build software?

- Building software includes (at least):
  - version control integration
  - identifying dependencies & their versions
  - configuring build commands for different build modes & environments
  - writing instructions for how to configure & build
  - test configuration & execution (performance & correctness)
  - automated code quality checking
  - scalable the compilation & linking
What does it mean to build software?

- Building software includes (at least):
  - version control integration
  - identifying dependencies & their versions
  - configuring build commands for different build modes & environments
  - writing instructions for how to configure & build
  - test configuration & execution (performance & correctness)
  - automated code quality checking
  - scalable the compilation & linking
  - possibly even deployment
What does it mean to build software?

- Building software includes (at least):
  - version control integration
  - identifying dependencies & their versions
  - configuring build commands for different build modes & environments
  - writing instructions for how to configure & build
  - test configuration & execution (performance & correctness)
  - automated code quality checking
  - scalable the compilation & linking
  - possibly even deployment

- It is the foundation of getting anything done.
What does it mean to build software?

- How many of you know how to *build* software?
  - Really.
What does it mean to build software?

- How many of you know how to build software?
  - Really.

- You should at least ask yourself:
  - What tools do you use?
  - What workflow?
  - What benefits do you get?
  - What are the painful points?
  - Why haven't you made them less painful?
What will be be using? CMake

- Cross-platform build management tool
- Used by large projects like KDE, Wireshark, LLVM, ...
What will be using?

- **CMake**
  - Cross-platform build management tool
  - Used by large projects like KDE, Wireshark, LLVM, ...

- **What does it do?**
  - Given a specification & configuration of your project, CMake creates the build commands for you
  - Analogous to autoconf (but easier to use)
What will be be using?

- CMake
  - Cross-platform build management tool
  - Used by large projects like KDE, Wireshark, LLVM, ...

- What does it do?
  - Given a specification & configuration of your project, CMake creates the build commands for you
  - Analogous to autoconf (but easier to use)

[DEMO]
What does this add?

- Why not just write makefiles manually?
What does this add?

- Why not just write makefiles manually?
  - May need different makefiles for different...
What does this add?

- Why not just write makefiles manually?
  - May need different makefiles for different
    - Operating Systems
    - Compilers
    - Libraries
    - Build Modes
    - ...
What does this add?

- Why not just write makefiles manually?
  - May need different makefiles for different
    - Operating Systems
    - Compilers
    - Libraries
    - Build Modes
    - ...
  - May need different source files for different “”
What does this add?

● Why not just write makefiles manually?
  – May need different makefiles for different
    • Operating Systems
    • Compilers
    • Libraries
    • Build Modes
    • ...
  – May need different source files for different “”
  – Specification can clearly capture
    • Libraries, versions, & even how to download them automatically
    • Semantics of compilation & how to use in analysis tools
What does this add?

- **Scalability**
  - Replace “make” with analogous scalable tools (“ninja”)
What does this add?

- **Scalability**
  - Replace “make” with analogous scalable tools (“ninja”)

- **Easier tool integration**
  - CMake can export compilation rules for other tools
What does this add?

- Scalability
  - Replace “make” with analogous scalable tools (“ninja”)
- Easier tool integration
  - CMake can export compilation rules for other tools

[DEMO]
Preliminary: Out of source builds

- A common bad habit is “in source” building
A common bad habit is “in source” building
  - Why is this bad?
Preliminary: Out of source builds

- A common bad habit is “in source” building
  - Why is this bad?
  - May need multiple builds at once: debug, release, ...
  - Pollutes version control
  - Makes clean builds complicated
Preliminary: Out of source builds

- A common bad habit is “in source” building
  - Why is this bad?
    - May need multiple builds at once: debug, release, ...
    - Pollutes version control
    - Makes clean builds complicated

- Use “out of source” builds instead
Using CMake

- CMakeLists.txt
  - A script in every directory of your project that controls how to build “things” in that directory
Using CMake

- CMakeLists.txt
  - A script in every directory of your project that controls how to build “things” in that directory

- Simple syntax
  - Case insensitive commands
    
    ```
    command( argument1 argument2 argument3 ...)
    ```
  - Let's revisit demo 1!
CMake allows you to specify targets

- Executables, libraries, "objects"

```cpp
add_executable(helloworld helloworld.cpp)
add_library(hellohelper hellohelper.cpp)
```
CMake allows you to specify targets
- Executables, libraries, “objects”

```cpp
add_executable(helloworld helloworld.cpp)
add_library(hellohelper hellohelper.cpp)
```

- And commands that can describe how to build those targets
  - Automatic for executable & library
    - `add_custom_command()` can build others
      - Documentation
      - Media
Directories

- Specify to look for build scripts in subdirectories

```
add_subdirectory(tools)
```
Directories

- Specify to look for build scripts in subdirectories
  
  \texttt{add_subdirectory(tools)}

- Specify search paths for header files or libraries
  
  \texttt{include_directories(include)}
  \texttt{link_directories(lib)}
Using libraries

- Specify that a specific target needs a library

```
target_link_libraries(target
    lib1 lib2 lib3 ...
)
```
Using libraries

- Specify that a specific target needs a library

  ```
  target_link_libraries(target
  lib1 lib2 lib3 ...
  )
  ```

- How might convenient library use affect program structure and design?
  - How might it help us begin to handle complexity?
General project management

- **Specifying project properties**
  - Define a project to access variables that control that project
    
```
    project (projectname)
    ```
General project management

- Specifying project properties
  - Define a project to access variables that control that project
    `project (projectname)`

- Print information out during the build process
  `message("Built with flags: \$\{CMAKE_CXX_FLAGS\}\")`
General project management

- Specifying project properties
  - Define a project to access variables that control that project
    `project (projectname)`

- Print information out during the build process
  `message("Built with flags: ${CMAKE_CXX_FLAGS}")`

- Controlling where things are built
  ```
  set (CMAKE_RUNTIME_OUTPUT_DIRECTORY
       "${PROJECT_BINARY_DIR}/bin")
  set (CMAKE_LIBRARY_OUTPUT_DIRECTORY
       "${PROJECT_BINARY_DIR}/lib")
  ```
General project management

- Finding a resource that you need to use
  
  ```
  find_package(externalproject)
  find_library(library)
  ```
General project management

- Finding a resource that you need to use
  
  ```
  find_package(externalproject)
  find_library(library)
  ```

- Installation
  
  ```
  install(TARGETS target1 target2 ...
  DESTINATION /tmp/
  )
  ```
Control structures

- IF

  { if(condition)
    elsif(condition2)
    else()
    endif()
Control structures

- **IF**
  
  ```
  if(condition)
  elseif(condition2)
  else()
  endif()
  
  foreach(loop_var arg1 arg2 ...) 
  command(${loop_var})
  endforeach(loop_var)
  ```

- **Looping**
  
  ```
  while(condition)...
  ```
Control structures

- **IF**
  
  \[
  \begin{align*}
  &\text{if}(\text{condition}) \\
  &\text{elsif}(\text{condition2}) \\
  &\text{else}() \\
  &\text{endif}()
  \end{align*}
  \\
  \begin{align*}
  &\text{foreach}(\text{loop\_var} \ \text{arg1} \ \text{arg2} \ \ldots) \\
  &\quad \text{command}(\${\text{loop\_var}}) \\
  &\quad \text{endforeach}(\text{loop\_var}) \\
  &\text{while}(\text{condition})
  \end{align*}

- **Looping**
  
  \[
  \begin{align*}
  &\text{function}(\text{function\_name} \ \text{arg1} \ \text{arg2} \ \ldots) \\
  &\quad \text{command}(\${\text{arg1}}) \\
  &\quad \text{endFunction}(\text{function\_name})
  \end{align*}

- **Functions**

\[
\begin{align*}
&\text{command}(\text{condition}) \\
&\text{else()} \\
&\text{endif()}
\end{align*}
\]
Examples

- Let's take a look at some examples....